Open Source streaming Media Server
with PHP front and backend
Open Source streaming Media Server with PHP front and backend

Name: Christian Peña Bello
Nationality: Spain
University: Universitat Politècnica de Catalunya (UPC)
Kind of study: Web development
Table of contents

TABLE OF CONTENTS ............................................................................................................. 3

SUMMARY .................................................................................................................................. 5

ACKNOWLEDGEMENTS ............................................................................................................ 6

INTRODUCING KAHO SINT-LIEVEN AND GHENT .............................................................. 7

KAHo SINT-LIEVEN ................................................................................................................. 7

GHENT .......................................................................................................................................... 8

DESCRIPTION OF THE PROJECT MISSION ............................................................................. 10

THEORETICAL PART ............................................................................................................... 12

STATE OF ART .......................................................................................................................... 12

CODIFICATION ......................................................................................................................... 12

CONTAINERS ............................................................................................................................ 14

DATABASES AND INDEXING .................................................................................................... 14

NETWORK PROTOCOLS ............................................................................................................ 15

VIDEO ON DEMAND TRANSMISSION .................................................................................... 16

STREAM GENERATION ............................................................................................................ 16

STREAM RECEPTION ................................................................................................................ 16

WEB INTERFACES ................................................................................................................... 17

DATABASES ............................................................................................................................... 19

SITE MAP ..................................................................................................................................... 25

PRACTICAL PART ..................................................................................................................... 28

SOFTWARE INSTALLATION .................................................................................................... 28

XAMPP ......................................................................................................................................... 28

MySQL ......................................................................................................................................... 28

CAKEPHP ..................................................................................................................................... 29

FFmpeg ......................................................................................................................................... 31

VIDEO STREAMING .................................................................................................................. 33

DATABASE .................................................................................................................................. 35

SITE DEVELOPING ................................................................................................................... 36

FEATURES OF THE APPLICATION ......................................................................................... 37
Summary

My name is Christian Peña and I am a Spanish exchange student coming from l’Escola d’Enginyeria de Terrassa (EET) in Terrassa, belonging to l’Universitat Politècnica de Catalunya (UPC). This is my final bachelor project and consists in the design of a Video on Demand Server and its interface. Although the research work began in my home university, most of the work has been done in KaHo Sint-Lieven, in Ghent.

The idea arose when Xavi Giró organized a group of students in order to do a big common project, or at least a group of project with something in common. The initial idea was to join an open source project and develop some part of it with the intention of participating in Google’s Summer of Code the following year. Later it became a big common project in which everybody took a part. The project consisted in developing a video on demand service, with open source software. Three projects were planned consisting in a server, and web and mobile device interfaces. My task was to develop the server and although, due to everybody’s later interests and chances, the common project was abandoned I carried on with my part.

I began my work during the summer of 2009 and carried on in the autumn semester. Here in Ghent I was given the chance of carrying on it by building a video on demand service for the http://www.ikdoeict.be including server and client part.
Acknowledgements

I would like to thank Xavi Giró who gave me the chance of beginning thinking about the project nearly a year ago, thanks to this I have developed a great interest about the subject. Albert Marquez, responsible of the TSC laboratory in Terrassa, who always tried to give a hand when I asked. And also Albert Roig, Pia Muñoz and Marcel Tella for making our meetings something funny and occasional help.

Here in Gent my greatest gratitude goes to Davy De Winne for giving me the chance of carrying on my work and giving a lot of facilities, good advices, help and, overall, freedom. Also the ICT teacher staff, as they have been always very kind and helpful.

And, at last, I would not like to forget all the people that have made my Erasmus experience as wonderful as it has been.
Introducing KaHo Sint-Lieven and Ghent

KaHo Sint-Lieven

The Katholieke Hogeschool Sint-Lieven is a young higher education institution in the East of Flanders with 4800 students and 500 employees. Although it is a young institution it has a rich tradition. The history of KaHo Sint-Lieven comes about in the histories of the 8 former higher education institutions, which have merged in 1995. Since the merger, KaHo Sint-Lieven has reorganized and grouped into 3 campuses in 3 regions in Aalst, Gent and Sint-Niklaas. Each campus has its own traditions and can look back on a rich – be it shorter or longer – past. Some of the eight founding institutions were established in the 19th century.

KaHo Sint-Lieven organizes bachelors, masters, postgraduates and short educations concerning the study-areas of biotechnology, health care, business studies, teacher training, industrial sciences and technology and nautical sciences.

I have been gladly surprised by the facilities it has and how well students are treated. From the very beginning we were given the chance to choose a project that fitted our interests and teachers took a lot of interest in it. We were given all the material we needed to develop it and teachers had a great interest in our work, always trying to give a hand when needed.

And not only with the project, but also as an exchange student, we were given accommodation near the university and were allowed to join all the courses we wanted to.

K.U.Leuven association

In accordance with the ideas of the Sorbonne (1999) and Bologna (2000) declarations about the restructuring of higher education in Europe, the Flemish Minister of Education in going through a process to reform the Flemish higher education system. One of the first objectives in this information process is to stimulate co-operation between institutions of higher education. Therefore universities and hogescholen have drawn up agreements in order to create “associations”, linking both types of institutions together. KaHo Sint-Lieven decided to associate with the Catholic University of Leuven (KU Leuven), Flanders’ largest and oldest university. On 11th July 2002 the association agreement between the KU Leuven and KaHo Sint-Lieven was signed.
The below institutions constitute the “Association KU Leuven”:

- Katholieke Universiteit Leuven
- Katholieke Universiteit Brussel
- Europese Hogeschool Brussel
- Hogeschool Sint-Lukas Brussel
- Hogeschool voor Wetenschap & Kunst
- Katholieke Hogeschool Brugge-Oostende
- Katholieke Hogeschool Kempen
- Katholieke Hogeschool Leuven
- Katholieke Hogeschool Limburg
- Katholieke Hogeschool Mechelen
- Katholieke Hogeschool Zuid-Westvlaanderen
- Lessius Hogeschool
- Groep T
- Katholieke Hogeschool Sint-Lieven

**Ghent**

Ghent is the capital of East Flanders, Belgium. The region was originally inhabited by the Celtics and Ghent come from the Celtic word ‘ganda’ that refers to the convergence of the Lys and Scheldt rivers, where Ghent lies.

During the middle Ages it was the second biggest city in northern Europe after Paris. Historically it has been a city that complained against high taxes and fought for their civil rights.

Nowadays is the Flemish city with the greater number of historical buildings, a rich cultural life, a great number of students and it is also very well located between Bruges and Brussels. With 237,250 inhabitants in the beginning of 2008, Ghent is Belgium’s second largest municipality by number of inhabitants.

Its official language is Flemish but people usually speak more languages, being English the most common of them.

The Ghent municipality includes the following villages: Afsnee, Desteldonk, Drongen, Gentbrugge, Ledeberg, Mariakerke, Mendonk, Oostakker, Sint-Amandsberg, Sint-Denijs-Westrem, Sint-Kruis-Winkel, Wondelgem y Zwijnaarde.

Ghent offers, forgetting its buildings, museums, shops, restaurants and an exciting nightlife with a lot of life music. It is easy to find life music every night or get lost in a bar.

It is great to discover the city by bike, the city is fully prepared and used to them, or just
get lost on foot.

Annually, during the second half of July takes place the Ghent festival, considered one of the biggest street festivals in the whole Europe.

It is a great place for students as there are a lot of them from whole Europe. It is impossible not to meet a lot of people and have fun. The atmosphere is great and it is possible to go out whenever you want.

From a Spanish point of view, the only bad thing is the weather, but I really expected it to be much worse. I was really surprised how people and the atmosphere of the city changed when spring arrived.
Description of the project mission

The goal of this project originally was to design and build a video on demand server using only open sources technologies. I would be using the Theora\(^1\) video encoder and the Vorbis audio encoder, all multiplexed in an Ogg container. I also had the intention of using VLC to control the video stream in the server and if needed, Icecast2. Finally neither VLC nor Icecast2 had been needed.

For the implementation of the server part I would be using PHP and MySQL\(^2\), while HTML5 and CSS would be used for the client part.

The video transcoding would be done with FFmpeg\(^3\) and the Apache server would be used. I chose the XAMPP packet, sometimes called LAMMP for the Linux version, which installs Apache, MySQL, PHP and Perl. All software would be running in the latest Ubuntu release.

During the project development I was suggested to use a PHP framework, CakePHP\(^4\), instead of PHP. The main reason was that I had no previous knowledge of PHP and the use of a framework would make the development easier and faster.

The idea of the project came with the release of HTML5 and the need of a free standard for the video on the Internet. The W3 suggested Ogg when the standard was released, with the Theora and Vorbis codecs, as the new video standard. Firefox, Chrome and Opera web browsers decided to give support to Ogg. Popular video on demand services such as YouTube and Daily Motion started to offer some videos using HTML5 with Ogg, but only as a test.

It can be said that it had not a great reception and had lot of detractors because other codecs had better performances. Other big companies like Apple did not agree with the W3 suggestion and proposed the proprietary H.264 that gives, if not the best, one of the best performances. Also Apple decided not to give support to Adobe’s Flash in their mobile devices, the, by far, most used format to stream video through the Internet. YouTube, owned by Google, is currently using H.264 to stream video to mobile devices as it is said that Flash is not a good solution for them.

However, by the time the project was first designed, during the summer of 2009 there was a big enthusiasm with Ogg and I decided to use as the streaming format. After a

\(^1\) [http://www.xiph.org](http://www.xiph.org)
\(^2\) [http://www.mysql.com/](http://www.mysql.com/)
\(^3\) [http://www.ffmpeg.org](http://www.ffmpeg.org)
\(^4\) [http://cakephp.org](http://cakephp.org)
few months this enthusiasm disappeared and seemed that the only true candidate to be the standard would be H.264.

The situation radically changed during May of the current year when Google opened the code of the VP8, after acquiring its creator, On2, and created the WebM\(^5\) project to give support to it. It has, by far, a better performance than Ogg and can compete with H.264, although this last still seems to have a slight better performance. Mozilla and Opera announced support to it, as well as Adobe and Microsoft, but this last not in a native way for the moment.

By now its not clear yet which format will become the standard but what can certainly be said is that Ogg has no future as an Internet standard. By default YouTube\(^6\) is still using Flash but it is possible to activate an HTML5 player. Then the videos will be streamed whether in WebM or H.264.

This is one of the main reasons that I decided to use H.264 as the first format instead of Ogg. But also, although it still can be used, the Ogg streaming was not as successful as I thought it would be. For the philosophy of this project I would prefer WebM but its release is too recent to implement it.

On the other hand the actual lack of agreement between companies brings a situation in which every browser only supports one of the main codecs and only Google’s Chrome is supporting both. This is why I finally decided to give support to both. First H.264 will be used, but if the browser is not supporting H.264, it will have the chance to play the video in Ogg.

The only requirement is that the browser supports HTML5, which is natively supported by all browsers except Microsoft Explorer that needs a plug-in until it’s the release of its 9\(^{th}\) version.

\(^5\) http://www.webmproject.org/
\(^6\) http://www.youtube.com/html5
Theoretical part

State of art
This chapter explains the most relevant technologies and software, adapted to the context, needed for the implementation of a video on demand server.

Codification
Digital codification consists in the translation of the analog electrical tension values that have been previously quantified to a binary system using established codes. Compression also takes place in this context, as it would be impossible to send video and audio with the current bandwidth available and no compression. The codec is the specific code that allows coding and decoding the data. Different codec cover different needs and are de reason why there are different formats and containers.

Video formats
MPEG\(^7\) (Moving Picture Expert Group) is a work group belonging to the ISO/IEC and formed by different companies and universities. Its function is to develop video and audio standards.

- MPEG-2 (part 2) or H.263 is a video codec for video broadcasting with standard TV quality. Used for modern TV services and DVDs.
- MPEG-4 (part 10), AVC or H.264 is a standard for high video compression. Its intention was to create a standard with good image quality and low bit rates, compared to older standards, while keeping simplicity in its design. It is considered one of the best formats and has great chances of becoming the most common format in the Internet during the following years.

VP8 is the last codec of On2 Technologies\(^8\). The recent acquisition, in 2009, of this company by Google has brought the liberation of its code and it is supposed to be the biggest rival for H.264. With its liberation, Google also announced the creation of the WebM project, which pretends to make VP8 the video standard in HTML5.

---

7 http://www.mpeg.org/
8 http://www.on2.com/
FLV is a proprietary format container. It is a variant of the H.263 standard called Sorenson Park and belongs to Adobe\(^8\). Nowadays, most of the video on the Internet is in FLV.

Theora is a free and open video compression format from the Xiph.org Foundation coming from the liberation of VP3’s code. It was called to be the future standard as it was the best open source video codec, but the recent liberation of VP8 have brought it to a second term.

H.264 and Ogg were chosen for this project. The first because is one of the candidates to become the standard video format in the coming years. And the second because has been, until now, the open source technology with more chances of becoming the standard. By the time this project is being written it would have been better to choose WebM, but its later release has been to close with the deadline of it.

**Audio formats**

**MPEG**

- MP3 (MPEG-1 layer 3) or H.260 is a digital lossy audio format. It was the first popular compression format due to its low rate that made possible the exchange of music files through the Internet. MPEG-2 audio introduced some improvements.

- MPEG-4 (part 3 or Audio) consists in different audio codifications lossy, lossless, for speech…

  In this context appears the Advanced Audio Coding, which bases its compression in perceptual models and has a good relation between its compression rate and quality. It was designed to substitute MP3 and has a better quality for the same bit rate.

Vorbis is a free and open audio compression format from Xiph.org Foundation. Its quality level is comparable to MPEG-2.

The audio format is not as important as the video as it represents a lower amount of data from the total. This is the main reason why it is chosen in function of the video format. H.264 goes with AAC, while Ogg uses Vorbis. WebM will also be using Vorbis.

\(^8\) [http://www.adobe.com/](http://www.adobe.com/)
Containers
Containers are a type of file which store, not code, information about the video, audio, subtitles, chapters, metadata and synchronism information following a format established in its specifications. All different kind of data is multiplexed inside the container.

- MPEG-1 (part 1) also known as PS (Program Stream) is not oriented for transmission as it has a low error protection.
- MPEG-2 (part 1) also known as TS (Transport Stream). TS offer a higher error protection so it is more oriented to transmission. The PS in this version is oriented to contain DVD streams.
- MPEG-4 (part 4), or mp4, was designed to transport audio and video and support a great number of codecs. It also supports subtitles or static images while allow Internet streaming.

SWF (Small Web Format) is a proprietary file format created by Macromedia and owned by Adobe. It is commonly used for animations and graphics but it is also used to broadcast video through the Internet using the Adobe Flash Player and FLV and MP3 as video and audio formats.

OGG is the container by Xiph.org Foundation and the native container for Theora, Vorbis and Speex (the voice codec by Xiph.org). It is also free and open and designed to give high efficiency in file compression.

The container is also a decision that depends on the choice of the video codec. The most common is using mp4 with H.264 and with Theora always Ogg.

Databases and indexing
A database is a group of data belonging to a common context and systematically stored for its later use. The software that controls the creation, maintenance and use of databases is called Database Management System (DBMS).

There are a lot of DBMS nowadays but for this project MySQL was used as it is one of the most popular, Open Source and comes with the XAMPP package.

In the computer context, indexing means to order registers using indexes. Indexing gives agility in searching, which is translated in faster queries. The data we index is called metadata.
Metadata means over the data, which at the same time means, that it is data describing other data. As in the video and audio formats and containers also exist standards for metadata like MPEG-7 (aka Multimedia Content Description interface) and MPEG-21 that defines a legitimate framework for sharing multimedia content, respecting the author rights and distribution.

**Network protocols**

The communication through a network is possible thanks to the TCP/IP model. TCP/IP is a layered model and takes the name of these protocols as they are the most relevant but they are not enough to handle everything and need more supporting protocols in other layers.

A layered model has the advantage of making easier designing protocols, as they have a concrete task. It also increments the interoperability between protocols, so they can be replaced for another depending on the needs.

These are the layers of TCP/IP:

The application layer communicates with the user and also controls the codification of the data (how the receiver have to receive the data so it can be understood) and the dialog (when is the turn to speak for each other). Some of the most important are the Hypertext Transfer Protocol (HTTP, used to exchange files that form web pages), DNS (Domain Network Service, that translates IP addresses to domain names) and FTP (File transfer Protocol, used to exchange files between systems).

The transport layer takes the information from the application layer and its function is to make it possible to communicate two applications running in different computers. Depending on our needs we can choose between two main protocols at this layer: TCP and UDP. Transmission Control Protocol (TCP) keeps more information about the network, called overhead, that make possible to recover from mistakes, resend lost or corrupted segments and use the link between hosts with high efficiency. It is used when we need to receive the data exactly in the same way we sent it, but with no time requirements. On the other hand there is User Datagram Protocol (UDP), it is much simpler is it ads no overhead and has no error control. This characteristics make it useful when we have time requirements and late resends are not useful, like sending video or voice.
There is also Real-time Transport Protocol (RTP) but it is not really a protocol itself. It is a complement as it adds some headers to the data that allow the video server to send data at the proper speed. These headers are packed in the UDP datagram.

The network layer makes the packet arrive to the correct host while trying to choose the best path to reach it. Here the Internet Protocol (IP) is the main protocol.

The link layer determines the way that hosts send and receive the data through a physical support given by the physical layer. Ethernet is the main protocol at this layer. The physical layer is the lowest one; it defines the physical media that will be used.

**Video on demand transmission**

The Video on Demand (VoD) consists in delivering a stored video when the user asks for it with no need to download the entire file before starting to play it. It differs from the real-time video in which the video is not stored.

**Stream generation**

To generate the stream a streaming server is needed. It defers from a regular server because it has to control the data in real time while the other just have to deliver the content. The streaming server will also have to maintain an interactive service with the user, who will be able to stop, pause, play or have a random access to the video. The streaming server will be able to deliver the stream in three different ways depending on the needs. If is sent to a unique user it is called unicast. If it is send to multiple users it will be called multicast. And if it is sent to all the nodes in a network it will be broadcast. Programs that are streaming servers or can act as it are VLC, Darwin Server, Quick Time Streaming Server or Flash Media Server.

**Stream reception**

Their function is to interpret the streaming data in order to play the files properly. Players can be a common software application or a plug-in in a web browser. The client will only need a player supporting video streams and the right codec to play the file. When the client asks for the video the server will start sending it. The client will wait a little bit just to store part of the video in a buffer, and when it considers it is
enough it will start playing it. If there is a punctual decrease in the speed transmission
the client will not notice, as it will still have part of the video saved in the buffer.
Nowadays, most of the media players support receiving streams from a network. However, with a web browser with HTML5 support there is no need for extra software as it is able to control the stream.

Web Interfaces
VoD needs an interface to be displayed and to allow the client interact with the video. HTML is the language that is used to write webs and is what the browsers interpret and then display. It allows the to combine text, images, sounds and videos. From the release of HTML5 in 2009, the inclusion of media is (or will be) something native in the browser that do not need from external plug-ins to be displayed.
On the other hand HTML has a big limitation: it is a static language. It is not capable of doing any mathematic operation or any interaction with the user. So if only HTML was used we would always have the same web page and a VoD service would not make any sense.

This limitation has made appear other additional and more complex languages that allow web pages to answer intelligently the client needs and the automation of some tasks. This other kinds of languages make possible dynamic webs. They generate the appropriate HTML content in function of the user requests.

It means that we have two kinds of languages: the ones that are interpreted by the client side and the ones that run in the server side.

Client side languages
HTML is a language that doesn’t need a server to be interpreted, but it is also limited when formatting a document. The solution is the introduction of CSS.
CSS works by defining with a special syntax the format of the web page. It can be applied to a whole web (meaning all its pages), to one page or just to one piece of HTML.
JavaScript is an object oriented scripting language used to access to objects in applications. It is mainly used in a web browser allowing the developing of improved user interfaces and dynamic web pages.
Both HTML and CSS are used in this project. JavaScript would have made a better interface, but due to the available time for making this project and my completely lack of knowledge about it was discarded.

**Server side languages**

PHP is one of the most used languages in the Internet because of its simplicity and power. It allows introducing the code mixed with the HTML and also a great quantity of functions to connect to databases.

There are also PHP frameworks. A web application framework is a software framework designed to support the development of dynamic websites, Web applications and Web services. Frameworks provide libraries for database access, templating frameworks and session management that make things easier when programming. They also promote code reusing.

CakePHP is an open source web application framework for producing web applications and is one of the most used. It is written in PHP and distributed with MIT License. It also encourages the use of the model-view-controller architectural pattern and has interesting tools like scaffolding, that relates database tables, and bake that generates standard code automatically with little interaction of the programmer.

This project was made with CakePHP as it allows developing webs very fast and easily with no need of knowing PHP deeply.

ASP is used with Windows servers. It is also a script language and is mixed with the HTML code.

Perl is a language that inherits some C characteristics. It has become a tool for web programming, database, bioinformatics or artificial intelligence.

Ajax is a language for the creation of interactive web applications. It is executed in the client side but a communication with the server is maintained in a second term. This allows changing a web page without the need of refreshing it. It would have also been a good tool but was discarded from the very beginning for the same reason as JavaScript.
Databases

Introduction to databases

A relational database was used for this project, which are the most commonly used. Relational databases are made up of relations, for these relations tables are used. A table has got columns or fields where different kind of data is stored and rows or entries where we have got the stored data. Then is easy to relate each piece of data, with any other piece that is in the same row, for example a name and a surname.

Each column in a table has a unique name and contains different data. We can choose the different type of data in any column, for example integers or characters, to fit our needs. Also we can specify that a field can or cannot be null or give it a default value if needed.

As said, each row represents an entry to a database table and because of the tabular format; each row will have the same attributes for each field. Rows an also be called records or tuples. Then each row will consist in a set of individual values that will correspond the columns.

In a relational database each record needs to be specified so we can indentify them uniquely. Intuitively we would identify a record by its name but as often names can be repeated it would not work. In this context keys appear. It is much more efficient and gives a better result to give each record a key rather than a name. Then each record will have a number identification that will make storing easier and will guarantee each record is unique. The field that acts as the ID is commonly called like this and it is artificially created and assigned to the record.

To guarantee the ID is unique we need to give it some attributes:

- Primary key: if we give this attribute to a field we are letting know the database that this field is the unique ID for this table.
- Auto increment: the DBMS automatically generates the ID number. The tool it uses to make it unique is an automatic increment. Then every time we introduce a new record, it will automatically increment the last ID number by adding a number we specify, by default one. By doing this we guarantee there are no equal IDs in a table.
- Not null: as it is a very important field in a table we have to make sure that it is not forgotten to fill it. If we give this attribute and also the auto increment, we
can completely forget about it when designing the interface with the entry as it will not be forgotten and also will not have a repeated value.

In a relational database there is the need to relate different tables, for example, in this project existed the need to relate videos with comments. In order to do it we would introduce a foreign key. The foreign key relates a record in a table with other records in another table.

Once we now how to relate tables, it is very important to know the existing relationships between tables. The relationship is defined by the number of elements in each side of the relationship. The relations can be:

- One-to-one: every row in a table can be linked to only another row in another table. This can also be implemented by adding the two tables so we keep the information in only one table. It is usually more useful to do it this way.
- One-to-many: every row in a table can be linked to many other rows in another table. To implement this a foreign key is added to the second table. Then we will be able to find the related rows by searching records with that specific foreign key.
- Many-to-many: every row in a table can be linked to many other rows in another table and at the same time, this other table can be linked to many other rows in the first table. The most efficient way to create this is adding an extra table in which foreign keys are stored in pairs, creating a relation, and also giving an extra ID to that relation.

For this project CakePHP conditioned the name of each key field. All table IDs are called ‘id’ and all foreign keys names refer to the table they come from.

**Database schema**

**Users**
The users table will keep information about the registered users of the service. Username is the assigned name for every user and is the field that will be displayed when referring to a user.

The role field has to do with the permissions of the user existing ‘user’, ‘teacher’ and ‘administrator’. A user who is not registered will only be able to watch videos, while a regular user will also have the chance of leaving comments in a video. A teacher will
be able to store videos and create channels and tags as well as administrate them, but only the ones owned by them. The administrator will have permissions to edit or delete everything.

It is also kept information about the real name of the user and an email direction in case it is needed to contact him or her.

The password is used for the login and it is stored encrypted in the database.

| id | int unsigned not null auto_increment primary key |
| username | varchar(50) not null |
| role | varchar(50) default 'user' |
| first_name | varchar(50) default null |
| last_name | varchar(50) default null |
| email | varchar(50) default null |
| password | varchar(50) default null |

**UNIQUE (username, email)**

Figure 1 Users table

**Videos**

The title, date of uploading and description, as well as the user_id (used to refer to the user who uploaded the video) is basic information about each video that will be displayed.

The url and auxurl are sources for the videos. In the first field paths to videos encoded with H.264 will be stored, while the second is an auxiliary field where the path to videos encoded with Theora will be stored.

In the thumbnail field we will keep the path to the generated thumbnail for each video.

All the videos and thumbnails will be stored in the server. As they are heavy files they cannot be stored in the database and results much more efficient to keep only their paths.

The user_id foreign key reflects the one-to-many relationship between users and videos. A user will be able to have many videos.
Comments are stored in a table as they will be light data. The date they were written and the user who did it is basic information that will often be displayed. The foreign keys user_id and video_id relate comments in a one-to-many relation with videos and users. While video_id allows knowing to which video does the comment belong, user_id let us know who was the author.

**Channels**

Channels are a way to organize videos. They have a many-to-many relationship with videos so we need a join table to implement it and this is the reason why no video_id is kept on this table.

The name of the channel is the only important information about the user while the user_id will let us know who created it.
Tags
Tags are another way to organize the videos so they can be sorted by topics. They work exactly the same way as channels do.

Join tables
These tables are used to implement the many-to-many relationship between channels and tags with users and videos.
Open Source streaming Media server

<table>
<thead>
<tr>
<th>TAGS_USERS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>int unsigned not null auto_increment primary key</td>
</tr>
<tr>
<td>tag_id</td>
<td>varchar(50) not null</td>
</tr>
<tr>
<td>user_id</td>
<td>int unsigned not null</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TAGS_VIDEOS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>int unsigned not null auto_increment primary key</td>
</tr>
<tr>
<td>tag_id</td>
<td>varchar(50) not null</td>
</tr>
<tr>
<td>video_id</td>
<td>int unsigned not null</td>
</tr>
</tbody>
</table>

Figure 6 Join tables
Site map

Home page

It is the default page in which appear the last added videos as well as some information user and the navigation menu.

In the top-right we have the user information, where if the user is logged can see its name. If the user is not logged in then there are two links, one for logging in and another to register.

In the left side there is the navigation menu. It is displayed by default in all pages as the user information. If an unregistered user or a user with the 'user' role are navigating through the website they will only have four links there:

- Home: redirects home
- Channels: lists links to all the available channels.
- Tags: lists links to all the available tags.
- Teachers: lists all the registered teachers.
If a user with the teacher role is at home, then he will have more options in the navigation menu:

- Add video: links to the new video form.
- My Channels: lists all his/her channels so he/she can access to them easily. It gives the chance to create a new channel, as well as editing and deleting the available ones.
- My Videos: lists all his/her videos so he/she can administer them easily as it also give the option to add, edit or delete a video.

And finally, if it is the administrator who is at home he will also have the chance of listing all the registered users. Information about all users will be displayed as well as the option of editing some of their information, or deleting them.

Channels, tags and teachers.

These are different ways of sorting the information but they all work in a similar way. They show the name of it and the videos related to them. It is the implemented way to find the videos easily avoiding having to look for them through the complete list. Also the administrator can use these lists to manage them, as the links for editing and deleting will be displayed for him.
This is the most important part of the website. Here the videos are embedded with a width of 400 pixels. The title, description, date of uploading and uploader are displayed here.

Comments will be displayed after the video in order of creation. They will also show the writer and the date and time they were written.

Registered users will also be able to comment the video as well as to edit or delete their own comments. The administrator will also have the permissions for doing this on this page.
Practical part

Software installation

**XAMPP**
Go to www.apachefriends.org and download the latest version, in this case 1.7.3a.
Then run the Ubuntu shell and login as root:

```
sudo -s
```

Extract the downloaded archive file to /opt directory:

```
tar xvzf xampp-linux-1.7.3a.tar.gz -C /opt
```

From now on we will be working in the /opt directory. Inside of it we will be able to see a folder named lampp, and inside of it, another folder named htdocs. In this folder there will be all the archives that we need to build our website.

To run lampp through the shell we will use this command, still as root, from the /opt/lampp directory:

```
./lampp start
```

To stop lampp we would just write stop instead of start.

**MySQL**
The installation of MySQL is totally completed when installing the XAMPP bundle. To run it we need to move to /opt/lampp/bin and type the command:

```
./mysql -h [host] -u [username]
```

Later, if we want to run MySQL as another user and have a password we would add -p in the end of the line, press Enter and the shell would ask for our password. For the moment, this will be enough.
CakePHP

As XAMPP is able to run PHP, we can install the CakePHP framework easily. Download the latest stable version, 1.3.0 in this case. The installation process is quite simple, it is only needed to rename the folder containing everything with our project name. In this case we will name our project vodserver.

```
mv /home/christian/Desktop/vodserver/ /opt/lampp/htdocs/
```

If Apache is running now we can check some information about our project state from the browser by writing in the direction bar localhost/webserver. Something similar to the following picture should appear:

![Figure 10 Mysql](image-url)
There are still some configuration parameters that should be change in order to make it work properly.

First we should make writable our tmp directory. After this CakePHP will still be giving a warning for the same reason, but referring to the cache folder, which is inside the tmp folder. For solving these two problems we will write the following lines in the bash:

```
chmod -R 777 /opt/lampp/htdocs/vodserver/app/tmp
chmod -R 777 /opt/lampp/htdocs/vodserver/app/tmp/cache
```

We are also asked to change the 'Security.salt' in order to add some security to the session variables. To do it we have to open the app/config/core.php file. Then locate line 198 and change the value for a similar one but with completely different characters. In line 203 we find the 'Security.cipherSeed' value, which also needs to be changed.

The database configuration is still not well configured. In it there should be the host, user, password, the name of the database and other values than allow CakePHP connecting to the database. We will find this archive in app/config/database.php.default, after changing the values we will have to change its name for database.php so CakePHP can recognize it has been changed.
There is no need to change the $test array as we won't be using it.

**FFmpeg**

Ubuntu distributions do not come with the necessary libraries to encode with H.264 and aac. So it is needed to install them when installing FFmpeg.

First it is necessary to uninstall x264, libx264-dev if they are installed because if they are it will not work. X264 is the open source implementation of H.264. We will write the following line in the bash:

```
sudo apt-get remove x264 libx264-dev
```

The following step is to get the necessary libraries to install ffmpeg and x264, but also, before the installation, it is good to install other repositories that may be needed:

```
sudo apt-get update
```
The following is to install the most current source files from the official x264 git repository, compile and install.

cd

git clone git://git.videolan.org/x264.git
cd x264
./configure
make
sudo checkinstall --pkgname=x264 --pkgversion "1:0.svn`date +%Y%m%d`+`git rev-list HEAD -n 1 | head -c 7`" --backup=no --default

We may also need to install libtheora, the library used to encode to Theora.

sudo apt-get install libogg-dev
cd
wget http://downloads.xiph.org/releases/theora/libtheora-1.1.1.tar.gz
tar xzvf libtheora-1.1.1.tar.gz
cd libtheora-1.1.1
./configure --disable-shared
make
sudo checkinstall --pkgname=libtheora --pkgversion "1.1.1" --backup=no --default

Then compile and install FFmpeg by getting the most current source file from the official FFmpeg SVN:

cd
svn checkout svn://svn.ffmpeg.org/ffmpeg/trunk FFmpeg
cd FFmpeg
./configure --enable-gpl --enable-version3 --enable-nonfree --enable-postproc --enable-pthreads --enable-libfaac --enable-libfaad --enable-
Video streaming

It is the key part of the project as the idea of the project came with the intention of streaming Ogg.

Before embedding the video in the web it looked a good idea to first stream a video to another computer by using VLC.

To stream a video with VLC we can use a graphical interface that makes work easier. We can see the streaming option by just opening file. Then click it and an assistant for the video streaming will be opened. We will be asked to choose the video source, then the protocol and the encapsulation. We can transcode the video before streaming but as this is inefficient was discarded from the very beginning and always tried to stream previously encoded videos.

VLC gives the option of sending a video stream with different protocols: UDP (unicast and multicast), RTP (unicast and multicast), MMS and HTTP. Then the encapsulation options, putting the encoded video in a container, depend on the compatibility with the protocol we have chosen.

First tests were made successfully by sending MPEG-2 video, just to make sure it worked. UDP was chosen because of its simplicity.

When trying the same but with Ogg it was found that there was no compatibility. UDP only allows sending TS (Transport Streams), and Theora and Vorbis cannot be encapsulated in that container.

By capturing some packets with Wireshark while receiving some YouTube and other video on demand services, I discovered that all of them are using HTTP. As it is a higher-level protocol, HTTP allows encapsulation with all formats.

However, during all time I have been developing this project I have not been able to stream Ogg properly. These are the unsuccessful experiments I tried:
- By using VLC streaming assistant and command line. But the most near to success experiment was when life transcoding, this way the audio worked and the video for nearly a second. After that the video was frozen and randomly, from time to time, it started playing again but with random speed.

- It was also tried to send the video, by Linux command line, encoding video to Ogg with ffmpeg2theora and connecting its output to a program called oggfwd that pulls the video to a video server. In this experiment icecast2 was used. This way of streaming also uses dvgrab, a program that takes the video from stream from a camera and sends it to the transcoder and it is used for live streaming. Avoiding this step is not a good solution probably because when transcoding directly a file there is no stream control. In other words, there is nothing that controls that the video is played at the correct speed and does not work.

- VLC also allows sending the video via icecast2 streaming server. I had major problems with its configuration but it also did not work. I have neither been able to embed a video stream in a HTML5’s video tag, as it only seems to work with a source file.

At this point it was decided to give up the Ogg streaming for a while and try to do it in a safe way. The VLC plug-in for Mozilla allows streaming a video file easily. It has some code to call it, some parameters and attributes. One of them, of course, is for the video source.

Until this time I had always thought that it was needed a stream controller in the server side and then the video stream had to be embedded. I even ignored all the HTML5 video embedding tutorials that did not use any middle software to control the stream. But I was completely wrong, as it is the browser who controls the video stream, and it is enough to write the path to the source file in the source attribute of the video tag.

One of the main reasons I did not realize is that I was always doing tests in a local network and with light video files, so they loaded really fast and started playing after having loaded. But when making tests with heavier files I realized that they already started playing before being completely loaded.

Later tests with Ogg videos were not completely successful as it begins playing before being fully loaded but do not allow random access to the video. However, tests with H.264 work perfectly.
**Database**

After installing XAMPP there are to ways to interact with MySQL: with an interface or through the console.

The graphical way is done through a web browser. To access to it we would open the browser, and type localhost. There we would see a XAMPP’s welcome page, in the left menu there is a link called PHPAdmin. By clicking it we would access to the MySQL graphical interface.

However, for this project the console was used as it is faster and we only need a few SQL statements to make it work. We can run it by typing this:

```
cd /opt/lampp/bin
./mysql -h [host] -u [username] -p
```

In case we do not create any user and run it in the localhost we can run it by only typing `./mysql`. The only condition is to have root permissions.

**Transcoding**

As videos will be uploaded in H.264 and ogv they first will need to be transcoded. For the transcoding FFmpeg will be used and the following command lines:

- **H.264 + aac**:
  ```
  ffmpeg -i mummies.flv -acodec libfaac -ab 96k -vcodec libx264 -s qvga -vpre slow -b 500k -bt 96k -threads 0
  /opt/lampp/htdocs/videos/mummies.mp4
  ```

- **theora + vorbis**:
  ```
  ffmpeg -i mummies.flv -acodec vorbis -vcodec libtheora -s qvga
  /opt/lampp/htdocs/videos/mummies.ogv
  ```

Transcoding H.264 and aac with ffmpeg hardly works and needs extra options. The most important are:

- `-acodec` is the paramater to choose the audio codec. Libfaac for aac and vorbis for vorbis.
- `-vcodec` allows us to choose video codec. Libx264 for H.264 and libtheora for Theora.
- `-s` specifies the size of the video. Qvga means 240x320, which is a good size to
fit the video in the website.
- -b refers to the bitrate.
- -bt sets the video bitrate tolerance. Specifies how far ratecontrol is willing to deviate from the target average bitrate value.
- -ab specifies the audio bitrate.

**Thumbnail generation**

Thumbnails are reduced-size versions of Pictures, used to help in recognizing and organizing them. In order to create them, we will take a reduced frame of the video stream.

FFmpeg can do it by giving it this order:

```
ffmpeg -itsoffset 4 -i mummies.flv -vcodec mjpeg -vframes 1 -an -f rawvideo -s 132x87 /opt/lampp/htdocs/thumbnails/mummeriesthumb.jpg
```

- -itsoffset indicates the number of seconds we will wait until we take the thumbnail.
- -vcodec is the codec used to code the new video, although it is just an image.
- -vframes indicates it is a video of one frame.

**Site developing**

The development of the site was fully made with CakePHP, a fully object oriented framework. It uses a MVC (Model, View, Controller).

In the Model the class is declared, the data validation is declared and it is declared the relation between this class and others. Using CakePHP’s name convention allows automatically connecting to the database and recognizing keys and foreign keys.

The Controller implements all the functions of the class and is the middle step with the model, that is closer to the database, and the view, that is closer to the interface.

The View is the interface to the user. It combines PHP with HTML and CSS. There it is also written all the specific code.

Cake has got a tool called Bake that automatically generates the models, basic controllers and basic views. Bake is a console application and has no graphical interface. In order to use it we have to move to the app/ folder and type the following:

```
./cake/console/cake bake
```
Open Source streaming Media server

Christian Peña Bello

There we can choose between many options but we only need to create models, controllers and views so we will choose, one by one these options. After having chosen what do we want to bake, Bake detects all the tables in the database and asks for what table do we want bake the model, controller or view. As it saves a lot of time, we will do it, one by one, with all of them.

Bake assists in the generation of the code letting you creating data validations, defining the relation between models, creating basic functions in the controllers...

After having baked all possible models, controllers and view we will have a simple application with a basic interface. The work then is to adapt that code to our needs by adding some more functionality.

Features of the application

Permissions

It allows administration of users by letting them register. Unregistered users have access to the content but cannot add any.

```php
function register(){
    if (empty($this->data)){
        $this->data['User']['password'] = md5($this->data['User']['password']);
        if ($this->User->save($this->data)){
            $this->Session->setFlash('Your registration information was accepted');
            $this->Session->write('user',$this->data['User']['username']);
            $this->redirect('http://localhost/vodserver/videos');
        } else{
            $this->data['User']['password'] = '';
            $this->Session->setFlash('There was a problem saving this information');
        }
    }
}
```

Figure 13 Register function

The standard registration assigns the role ‘user’ that can be later changed by the administrator to ‘teacher’. A registered user will have the chance of writing comments to the videos, as well as the option of editing or deleting them.

When users are upgraded to teachers they will be allowed to add videos, channels and tags, as well as editing and deleting them.

The administrator will have all before permissions, but for every user. While users and teachers only are allowed to administer their content, the administrator will have permissions for editing and deleting everything.

Also they will be able to administrate users by upgrading them to teachers, deleting
them or just change some of their information.

```php
<?php if ($userRole == $teach || $userRole == $admin) { ?>
    <li><?php echo $this->html->link($this->Html->link(sprintf(__('New %s'), tr($this))), 'channels'); ?></li>
    <li><?php echo $this->html->link($this->Html->link(sprintf(__('My Videos'), tr($this))), 'my-videos'); ?></li>
    <li><?php echo $this->html->link($this->Html->link(sprintf(__('My Channels'), tr($this))), 'my-channels'); ?></li>
<?php } ?>
</li>
<?php }
```

**Figure 14 Permissions administration**

### Channels and tags

The application allows the creation of channels and tags as ways of sorting the videos. It can also sort the videos by the teacher who uploaded them.

A teacher can have many channels and put videos in them. A user will have the chance of listing all the channels, and once a channel is chosen all videos in it will be listed.

The teacher and the administrator also will have the chance to list their own channels so it is easier to administer them.

By default, when adding a video the application asks if it is belonging to a video and allows to choose from the available channels. When adding from a channel the application automatically adds the video to that channel.

When adding a video, it can be added to more than one channel and it is possible to add it to more channels by editing the video information.

Tags work in the same way, but although the application keeps information about its creator, they are not supposed to belong to anyone, so it is not possible to list tags by user.

The function `listTeachers`, placed in the `users_controller` allows all users to explore videos by the teacher who uploaded it.

```php
function listTeachers(){
    $this->set('teachers', $this->User->find('all', array('conditions' => array('role' => 'teacher'))));
}
```

**Figure 15 listTeachers function**
Watching videos

Videos are listed by showing their thumbnail, a link to the video, their basic uploading information and their description.

When we click a video link we are redirected the video view file. In there we have the necessary HTML5 code to watch the videos in two formats H.264 or Ogg, depending on our browser’s compatibility.

![Video tag](image)

Comments belonging to each video are listed just after the video giving all registered users the option to add one. If this link is clicked we will be redirected to the add comment page.

Lacks of the application

The most important lack is that it is not possible to remotely upload a video to the server. Tries were not successful and the closer achievement was to write the data of the file to the database but not uploading the file.

In order to make it possible, values of the php.ini, the file in the XAMPP directory where configuration information about PHP is saved, were changed. These values were the max_upload_file, post_max_size, and max_execution_time. The first two have to do with the maximum size it is allowed to upload, and were given a value of 1GB. The last parameter is about the maximum amount of time it can be working, a value of an hour was given.

However, the problem had to with CakePHP and my lack of knowledge about it, as I was not able to integrate a plug-in for uploading in order to make it work.

Another important lack is the frontend usability. It does not have a lot of options and a lot of little things could be improved to make the frontend more comfortable and error free.
Conclusion

When I began this project I had a very poor background of all the used technologies. So it can be said that everything was new for me. I also have developed such a great interest in web development and their technologies.

For the streaming part everything was new for me except the network protocols. I really enjoyed testing with formats, their compatibilities with containers and protocols as well as reading about their current situation.

My programming background was not also very good as it is not one of the most important things in my home university study plan. Fortunately I knew enough about classes that allowed me to understand enough CakePHP. As my intention in the beginning was to program with PHP I also was learning it also with MySQL during the first weeks.

One thing I also enjoyed is getting fluent with Ubuntu’s console, solving easily problems that a year ago would have taken me a lot of time. Solving Ubuntu’s problems with libraries, configuration of programs and making them work by command line have also been a great headache, but most of them were solved successfully.

However, there are a lot of things to improve, beginning for the file uploading. But not only this, also channel and tag manager could be improved. As it is implemented now they are too similar, and adding tags to a video or a video to a channel is not easy enough. Also listing things in groups of ten instead of listing the whole amount of available videos, tags and channels.

Also many interesting features could be added, like a better integration with the http://www.ikdoeict.be by using its users database, also its layout is not the most appropriate for watching videos, as it does not occupies the whole screen and, as a result, the video has to be watched in a low resolution. Watching the video in a full screen mode is a problem of the browser’s video player, as the HTML5 does not have this option.

Adding a searcher would make the search of videos more comfortable by far.
Storing videos in more than one resolution would also be an interesting feature. It would allow watching videos with high resolution in a computer but also at the same time in low resolution for mobile devices.

Adding some JavaScript and also a deeper work with HTML5 would improve the interface, as it has a lot of possibilities.

But maybe the most important would be to change Ogg for WebM that will be possible in a few months as it is beginning to be possible to transcode video to this format and web browsers are beginning to be compatible with it.
List of resources

- [http://www.wikipedia.org](http://www.wikipedia.org) (Catalan, Spanish and English version)
- [http://www.genbeta.com](http://www.genbeta.com)
- [http://www.muylinux.com](http://www.muylinux.com)
- [http://www.webmonkey.com](http://www.webmonkey.com)
- [http://www.neoteo.es](http://www.neoteo.es)
- [http://forum.videolan.org](http://forum.videolan.org)
- [http://www.videolan.org/doc](http://www.videolan.org/doc)
- [http://www.ffmpeg.org](http://www.ffmpeg.org)
- [http://www.w3schools.com](http://www.w3schools.com)
- [http://www.apachefriends.org](http://www.apachefriends.org)
- [http://ubuntu.forums.org](http://ubuntu.forums.org)
- [http://www.cakephp.org](http://www.cakephp.org)
- [http://www.xiph.org](http://www.xiph.org)
- [http://en.flossmanuals.net/theoracookbook](http://en.flossmanuals.net/theoracookbook)
- [http://www.icecast.org](http://www.icecast.org)
- Cisco’s Networks fundamentals version 4.0 English
- “Servidor web de video sota demanda basat en Videolan” by David Vera
- “Usability improvements on a Metadata Server for Video on Demand based on Free Software” by Laurens de Vos.
- “Servidor de Corrents d’audio i vídeo bastat en programari i formats lliures” by Àngela Abad.
- "DVB-T Measurements with Promax TV Explorer and analysis of DVB Transport Streams" by Anna Arias.
- “Beginning CakePHP: From novice to professional” by David Golding (Apress, 2008)